String vs StringBuilder vs StringBuffer in Java — What’s the difference and when to use each

In Java there are 3 main classes you commonly deal with when working with text: `String`, `StringBuilder`, and `StringBuffer`.

Here’s how they differ and when each is appropriate.

📌 Characteristics & Differences

Feature / Class	`String`	`StringBuilder`	`StringBuffer`
Mutability	Immutable — once created, you cannot change the content. Operations like concatenation/replace return a new String.	Mutable — you can modify the content (append/insert/replace /delete) without creating new objects.	Mutable — like StringBuilder, allows in-place modifications.
Thread safety (synchronization)	Thread-safe by nature (immutable).	Not synchronized → *not thread-safe*.	Synchronized → *thread-safe*.
Performance (for modifications / concatenation loops)	Poor — each modification creates new object, causing overhead and garbage collection.	Fastest — minimal object creation and overhead.	Slower than StringBuilder because of synchronization overhead.
Memory / Interning / Storage	`String` literals are interned (string pool), so reuse is possible and memory optimized. Good for constants, fixed text.	Resides on heap; no string pool — typical object.	Also on heap.